PRESIDENCY UNIVERSITY

GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS

Private University Estd. in Karnataka State by Act No. 41 of 2013

PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

Project Title

GROVER'S ALGORITHM

Course Name: Quantum Computing

Course Code: CSE 3080

Section: 6 CAI-3

Faculty Name: Dr.Jayakumar V

Team Members

Vishal D Hanchate (20201CAI0192)

Vikas N M (20201CAI0193)

Sachin R (20201CAI0166)

Naveen R (20201CAI0163)

Vivan Sanjay Pothala (20201CAI0180)

Introduction

Quantum theory is a revolutionary advancement in physics and chemistry that emerged in the early twentieth century. It is an elegant mathematical theory able to explain the counterintuitive behavior of subatomic particles, most notably the phenomenon of entanglement. In the late twentieth century it was discovered that quantum theory applies not only to atoms and molecules, but to bits and logic operations in a computer. This realization has brought about a revolution in the science and technology of information processing, making possible kinds of computing and communication hitherto unknown in the Information Age.

Our everyday computers perform calculations and process information using the standard (or classical) model of computation, which dates back to Turing and von Neumann. In this model, all information is reducible to bits, which can take the values of either 0 or 1. Additionally, all processing can be performed via simple logic gates (AND, OR, NOT, XOR, XNOR) acting on one or two bits at a time, or be entirely described by NAND (or NOR). At any point in its computation, a classical computer's state is entirely determined by the states of all its bits, so that a computer with bits can exist in one of possible states, ranging from to.

The power of the quantum computer, meanwhile, lies in its much richer repertoire of states. A quantum computer also has bits - but instead of 0 and 1, its quantum bits, or qubits, can represent a 0, 1, or linear combination of both, which is a property known as superposition. This on its own is no special thing, since a computer whose bits can be intermediate between 0 and 1 is just an analog computer, scarcely more powerful than an ordinary digital computer. However, a quantum computer takes advantage of a special kind of superposition that allows for exponentially many logical states at once, all the states from to. This is a powerful feat, and no classical computer can achieve it.

The vast majority of quantum superpositions, and the ones most useful for quantum computation, are entangled. Entangled states are states of the whole computer that do not correspond to any assignment of digital or analog states of the individual qubits. A quantum computer is therefore significantly more powerful than any While today's quantum processors are modest in size, their complexity grows continuously. We believe this is the right time to build and engage a community of new quantum learners, spark further interest in those who are curious, and foster a quantum intuition in the greater community. By making quantum concepts more widely understood - even on a general level - we can more deeply explore all the possibilities quantum computing offers, and more rapidly bring its exciting power to a world whose perspective is limited by classical physics.

With this in mind, we created the IBM Quantum Composer to provide the hands-on opportunity to experiment with operations on a real quantum computing processor.
This field guide contains a series of topics to accompany your journey as you create your own experiments, run them in simulation,

If quantum physics sounds challenging to you, you are not alone. But if you think the difficulty lies in "hard math", think again. Quantum concepts can, for the most part, be described by undergraduate-level linear algebra,

The true challenge of quantum physics is internalizing ideas that are counterintuitive to our day-to-day experiences in the physical world, which of course are constrained by classical physics. To comprehend the quantum world, you must build a new

The true challenge of quantum physics is internalizing ideas that are counterintuitive to our day-to-day experiences in the physical world, which of course are constrained by classical physics. To comprehend the quantum world, you must build a new intuition for a set of simple but very different (and often surprising) laws.

The counterintuitive principles of quantum physics are:

A physical system in a definite state can still behave randomly.
Two systems that are too far apart to influence each other can nevertheless behave in ways that, though individually random, are somehow strongly correlated.

Unfortunately, there is no single simple physical principle from which these conclusions follow - and we must guard against attempting to describe quantum concepts in classical terms! The best we can do is to distill quantum mechanics down to a few abstract-sounding mathematical laws, from which all the observed behavior of quantum particles (and qubits in a quantum computer) can be deduced and predicted.

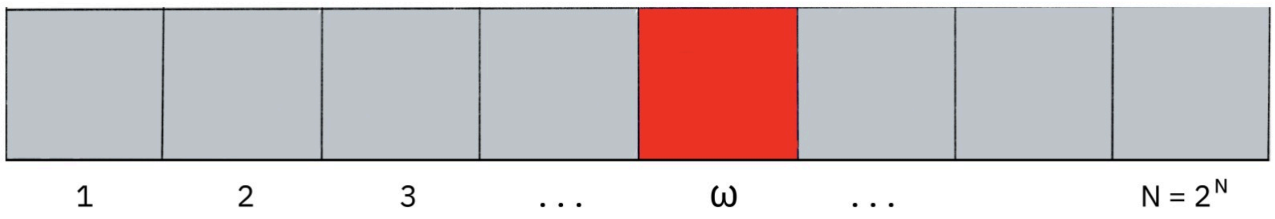
Keep those two counterintuitive ideas in the back of your mind, let go of your beliefs about how the physical world works, and begin exploring the quantum world.

Grover's algorithm

We are now in a good place to discuss our first quantum algorithms and subroutines. Let's begin with Grover's search algorithm and the *amplitude amplification trick*.
You have likely heard that one of the many advantages a quantum computer has over a classical computer is its superior speed searching databases. Grover's algorithm demonstrates this capability. This algorithm can speed up an unstructured search problem quadratically, but its uses extend beyond

that; it can serve as a general trick or subroutine to obtain quadratic run time improvements for a variety of other algorithms. This is called the *amplitude amplification trick*. But before we start the simulations, let’s look at the unstructured search problem.

Suppose you are given a large list of I items. Among these items is one item with a unique property that we wish to locate. We will call this one the winner, w. Think of each item in the list as a box of a particular color. Say all items in the list are gray except the winner w, which is red.



To find the red box - the marked item - using classical computation, one would have to check on average N/2 of these boxes, and in the worst case, all N of them. On a quantum computer, however, we can find the marked item in roughly N steps with Grover's amplitude amplification trick. It was proven (even before Grover's algorithm was discovered!) that this speedup is in fact the most we can hope for [Bennett,
1997 L} 1. A quadratic speedup is indeed a substantial time-saver for finding marked items in long lists. Additionally, the algorithm does not use the list's internal structure, which makes it generic; this is why it immediately

Amplitude amplification

So how does the algorithm work? Before looking at the list of items, we have no idea where the marked item is. Therefore, any guess of its location is as good as any other, which can be expressed in terms of a quantum state called a uniform superposition:

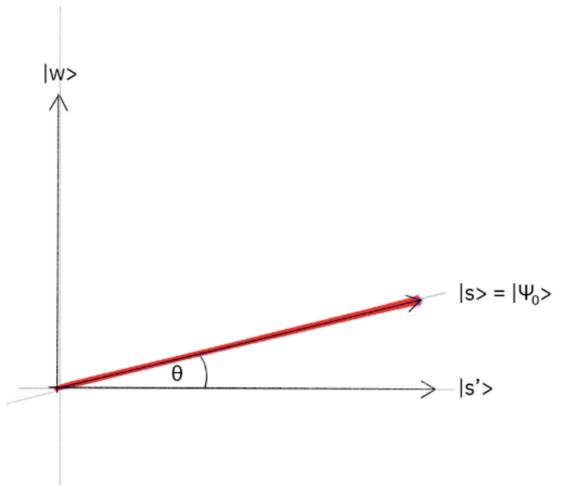
$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle.$$

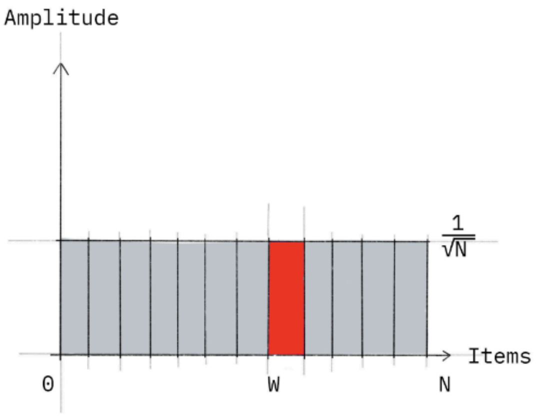
If at this point we were to measure in the standard basis {x)}, this superposition would collapse, according to the fifth quantum law, to any one of the basis states with the same probability of 1/n=1/2^n. Our chances of guessing the right value W is
therefore 1 in 2", as could be expected. Hence, on average we would need to try
about I = 2" times to guess the correct item.

Enter the amplitude amplification procedure, which is how a quantum computer significantly enhances this probability. This procedure stretches out (amplifies) the amplitude of the marked item, which shrinks the other items' amplitudes, so that measuring the final state will return the right item with near certainty.

This algorithm has a nice geometrical interpretation in terms of two reflections, which generate a rotation in a two-dimensional plane. The only two special states we need to consider are the winner w) and the uniform superposition S. These two vectors span a two-dimensional plane in the vector space C. They are not quite perpendicular because w) occurs in the superposition with amplitude IN 1/2 as well. We can, however, introduce an additional state s') that is in the span of these two vectors, is perpendicular to w), and is obtained from s) by removing w and rescaling.

Step 1 The amplitude amplification procedure starts out in the uniform superposition s. (The uniform superposition is easily constructed from [s) = H®n 0)n, as was shown in Creating superpositions and quantum interference.)





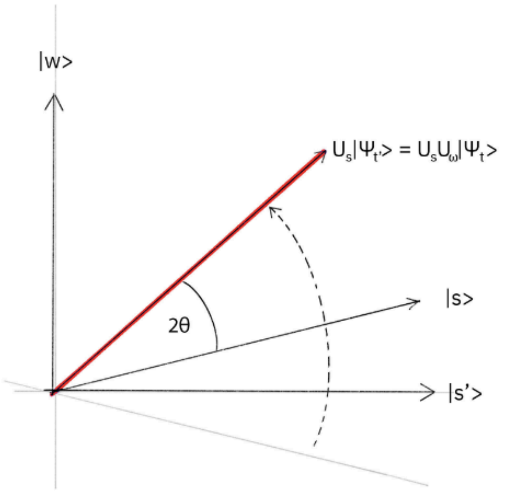
The left graphic corresponds to the two-dimensional plane spanned by perpendicular vectors (w) and s'), which allows us to express the initial state as (s) = sinfw) + costls'), where 0 = arcsin(slw) = aresin 1/rootn The right graphic is a bar graph of the amplitudes of the state s).

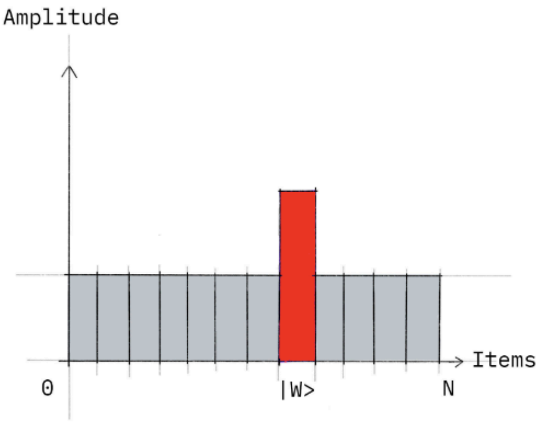
Step 2 We apply the oracle reflection Uf to the state s).

Geometrically this corresponds to a reflection of the state Is about Is'). This transformation means that the amplitude in front of the w) state becomes negative, which in turn means that the average amplitude (indicated by a dashed line) has been lowered.

Step 3 We now apply an additional reflection Us about the state s):

Us = 2Is) (sl - 1. This transformation maps the state to UsUf Is) and completes the transformation.

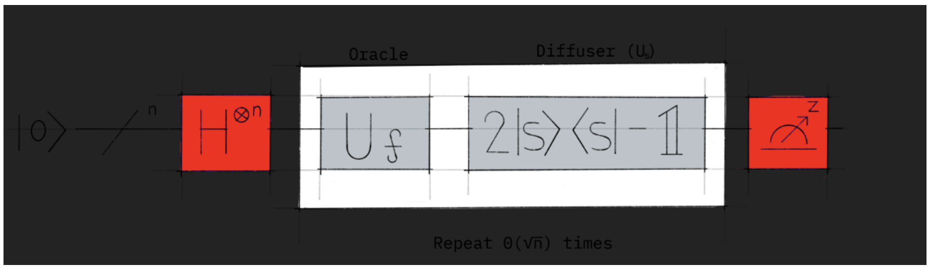




Two reflections always correspond to a rotation. The transformation UsU rotates the initial state Is' closer toward the winner w. The action of the reflection Us in the amplitude bar diagram can be understood as a reflection about the average amplitude. Since the average amplitude has been lowered by the first reflection, this transformation boosts the negative amplitude of w) to roughly three times its original value, while it decreases the other amplitudes. We then go to Step 2 to repeat the application. This procedure will be repeated several times to focus in on the winner.

After t steps, the state will have transformed to lat), where (r) = (U, Ur) Is).
How many times do we need to apply the rotation? It turns out that roughly VI rotations suffice. This becomes clear when looking at the amplitudes of the state ?)We can see that the amplitude of w) grows linearly with the number of applications ~ tN-1/2 However, since we are dealing with amplitudes and not probabilities, the vector space's dimension enters as a square root. Therefore it is the amplitude, and not just the probability, that is being amplified in this procedure.

In the case that there are multiple solutions, M, it can be shown that roughly y rotations will suffice.



Statement of the problem

Any search task can be expressed with an abstract function f(x) that accepts search items a. Ifthe item a is a solution to the search task, then f(x) = 1. If the item a isn't a solution, thenf(x) = 0. The search problem consists of finding any item xo such that f(x) = 1.

The task that Grover's algorithm aims to solve can be expressed as follows: given a classical function f(x) : {0, 1}n - {0, 1}, where n is the bit-size of the search space, find an input x0 for which f(x) = 1. The complexity of the algorithm is measured by the number of uses of thefunction f(x). Classically, in the worst-case scenario, f(x) has to be evaluated a total of N - 1times, where N = 2" trying out all the possibilities. After I - 1 elements, it must be the last element. Grover's quantum algorithm can solve this problem much faster, providing a quadratic speed up. Quadratic here implies that only about N evaluations would be required, compared to N.

Understanding Grover's Algorithm

At its core, Grover's algorithm aims to solve the problem of searching an unstructured database with N elements in a time complexity of O(√N), in contrast to classical algorithms which typically require O(N) time. This quadratic speedup has immense implications for a wide range of applications, particularly in fields where extensive search operations are involved.
The algorithm operates by employing a combination of amplitude amplification and phase inversion. It starts with an equal superposition of all possible inputs, represented by a quantum state. Then, a so-called "oracle" is used to mark the desired solution(s) in the search space, making them distinguishable from the others. The oracle flips the phase of the marked elements, effectively creating a phase difference between the marked and unmarked states.

Next, the algorithm applies a reflection operator, commonly referred to as the Grover diffusion operator, to amplify the amplitudes of the marked states while suppressing the amplitudes of the unmarked states. This amplification process enhances the probability of measuring the marked states upon measurement, thereby increasing the likelihood of finding the solution(s) to the search problem.
The algorithm repeats the oracle and amplification steps $\sqrt{N}$ times, leading to a quadratic speedup in comparison to classical algorithms. Once the solution(s) are found, they can be measured and extracted, providing the desired outcome of the search problem.

Applications and Implications

Grover's algorithm holds immense potential for a variety of practical applications. Some notable use cases include:

1. Database Search: Grover's algorithm enables faster searches in unsorted databases, providing an advantage in scenarios where finding specific information is crucial. This has implications for areas such as data mining, information retrieval, and cryptography.
2. Optimization Problems: Many real-world problems involve optimizing a specific function or searching for the global minimum or maximum. Grover's algorithm can offer speedups in solving optimization problems, leading to more efficient resource allocation, logistical planning, and financial modeling.
3. Machine Learning: Grover's algorithm can be employed in machine learning tasks such as clustering, classification, and recommendation systems. It has the potential to enhance the efficiency of training models, improving predictive accuracy and reducing computational complexity.
4. Graph Theory: Grover's algorithm can be utilized to search through graphs, facilitating applications in network analysis, route planning, and social network analysis. The ability to find the shortest path or identify key nodes efficiently has far-reaching implications.

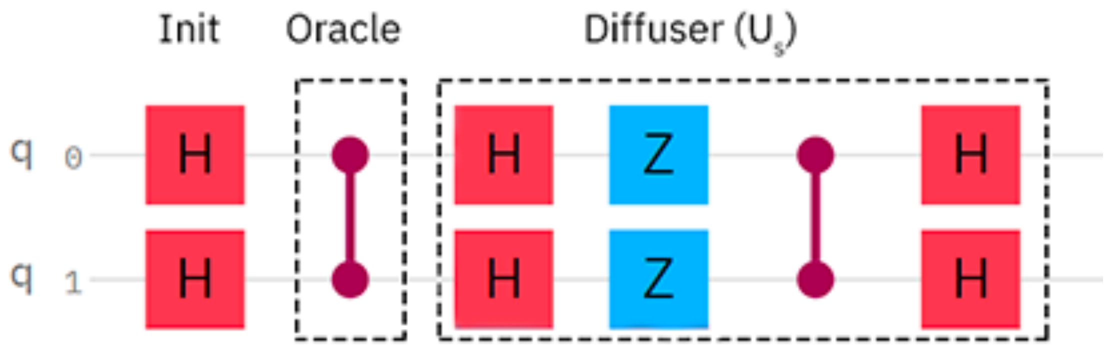
Challenges and Future Directions

While Grover's algorithm has revolutionized quantum search, it is not without its limitations. One major challenge lies in the physical implementation of quantum systems with the required coherence and precision to execute the algorithm effectively. Current quantum computers suffer from various sources of noise and errors, which can impact the success rate and scalability of Grover's algorithm.
Furthermore, the applicability of Grover's algorithm is restricted to unstructured search problems, limiting its scope in certain domains. Other quantum algorithms, such as Shor's algorithm for prime factorization, may be better suited for specific tasks.
In the coming years, advancements in quantum hardware and error correction techniques are likely to address some of these challenges. As quantum computing technology continues to evolve, Grover's algorithm

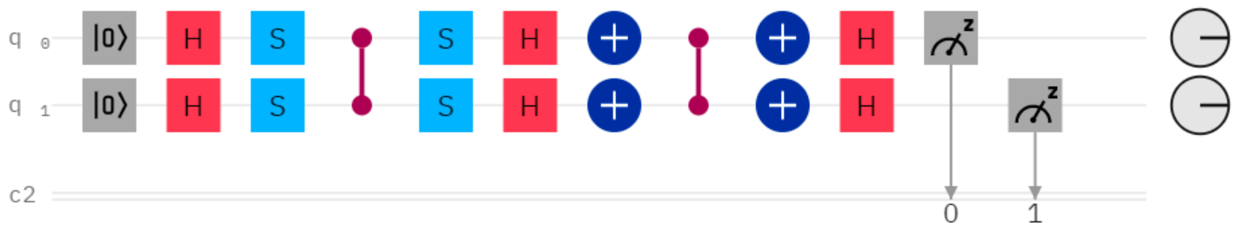
EXAMPLE

Full circuit for |w> = |11>

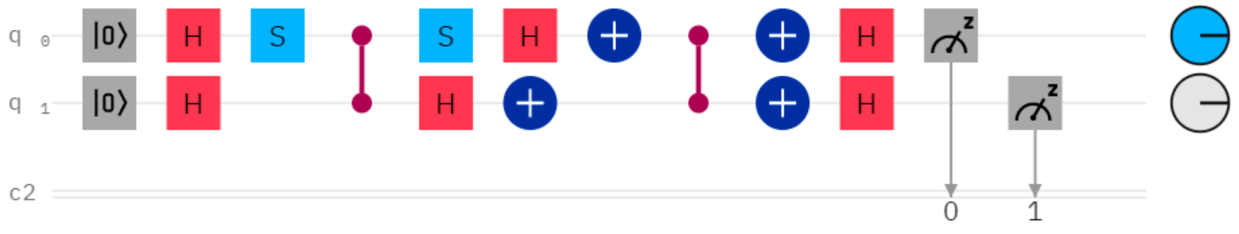
Since in the case of N=4, only one rotation is required, we can combine the above components to build the full circuit for Grover's algorithm for the case |w> = |11>



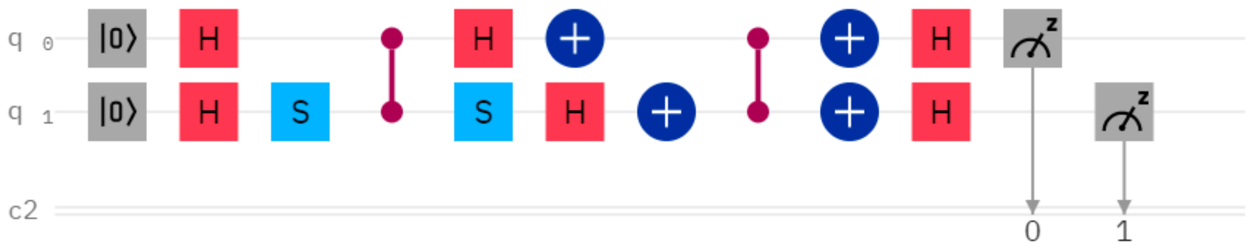
Grover |w> = |00>



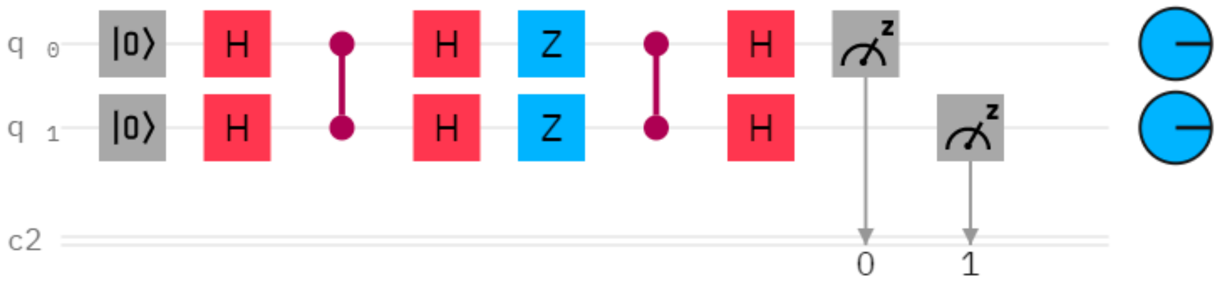
Grover |w> = |01>



Grover |w> = |10>



Grover |w> = |11>



CODE

```
# Import necessary libraries
from qiskit import QuantumCircuit, execute, Aer

def grover_algorithm(n):
    # Create quantum circuit with n qubits and n classical bits
    circuit = QuantumCircuit(n, n)

    # Apply Hadamard gates to all qubits
    circuit.h(range(n))

    # Oracle: Mark the solution (here, we mark the state 111)
    circuit.barrier()
    circuit.x(range(n))
    circuit.h(n-1)
    circuit.mct(list(range(n-1)), n-1)
    circuit.h(n-1)
    circuit.x(range(n))
    circuit.barrier()

    # Amplification: Repeat oracle and diffusion steps
    for _ in range(int(round((3.14/4) * (2**(n/2))))):
        circuit.barrier()

        # Apply oracle
        circuit.x(range(n))
        circuit.h(n-1)
        circuit.mct(list(range(n-1)), n-1)
        circuit.h(n-1)
        circuit.x(range(n))

        # Apply diffusion operator
        circuit.h(range(n))
        circuit.x(range(n))
        circuit.h(n-1)
        circuit.mct(list(range(n-1)), n-1)
        circuit.h(n-1)
        circuit.x(range(n))
        circuit.h(range(n))

        circuit.barrier()

    # Measure all qubits
    circuit.measure(range(n), range(n))

    return circuit

# Set the number of qubits
num_qubits = 3

# Create Grover's algorithm circuit with the given number of qubits
grover_circuit = grover_algorithm(num_qubits)

# Simulate the circuit using the Qiskit Aer simulator
simulator = Aer.get_backend('qasm_simulator')
job = execute(grover_circuit, simulator, shots=1000)

# Get the results
result = job.result()
counts = result.get_counts()

print("Measurement results:")
print(counts)
```

OUTPUT:

```
Measurement results:
{'000': 0, '001': 0, '010': 0, '011': 0, '100': 0, '101': 0, '110': 0, '111': 1000}
```